Amendments to the claims,
Listing of all claims pursuant to 37 CFR 1.121(c)

This listing of claims will replace all prior versions, and listings, of claims in the application:

1. (Currently amended) A method for dynamically generating program code adding behavior to a program based on attributes, the method comprising:

adding a static field of type Component ~~component object~~ to a program class of the program to create a component;

defining at least one attribute specifying declaratively behavior to be added to the program, wherein said at least one attribute comprises active metadata used to generate program code for inclusion in the program;

associating said at least one attribute with the component; and

in response to instantiation of the component at runtime, generating a subclass based on the program class and said at least one attribute, the subclass including dynamically generated program code based on said at least one attribute.

2. (Original) The method of claim 1, wherein said defining step includes defining a particular attribute using active metadata, so as to provide a mechanism for generation of program code from said particular attribute.

3. (Original) The method of claim 2, wherein said active metadata dynamically generates code for inclusion in a subclass based on the program class.

4. (Original) The method of claim 1, wherein the generating step includes generating a subclass comprising an instance of a declared component class.

5. (Currently amended) The method of claim 1, wherein the generating step includes generating a subclass comprising an instance of ~~an abstract~~ component class declared as abstract.

6. (Original) The method of claim 1, wherein said defining step includes defining at least one attribute based on comments in source code of the program class.

7. (Original) The method of claim 6, further comprising:

precompiling a class containing static attributes from said comments.

8. (Original) The method of claim 7, further comprising:

loading the class containing static attributes before subclass generation when a component is instantiated.

9. (Original) The method of claim 6, further comprising:

defining an automated mapping between attribute syntax in comments and attribute syntax as expressed in generated program code.

10. (Original) The method of claim 1, wherein said defining step includes defining at least one attribute in a property file external to the program class.

11. (Original) The method of claim 10, further comprising:

compiling a class containing dynamic attributes from said property file.

12. (Original) The method of claim 11, further comprising:

loading the class containing dynamic attributes before subclass generation when a component is instantiated.

13. (Original) The method of claim 10, further comprising:

defining an automated mapping between attribute syntax in a property file and attribute syntax as expressed in generated program code.

14. (Original) The method of claim 10, wherein attributes in the property file comprise property name and property value pairs.

15. (Original) The method of claim 1, wherein said defining step includes defining attributes for a superclass from which the program class inherits.

16. (Original) The method of claim 1, wherein said defining step includes defining attributes for the program class' package from which the program class inherits.

17. (Original) The method of claim 1, wherein said defining step includes defining attributes for an interface from which the program class inherits.

18. (Original) The method of claim 17, wherein said generating step includes generating an instance of a subclass to mock the behavior of the interface.

19. (Original) The method of claim 1, wherein said generating step includes generating code for a non-abstract method body based on an attribute defined for an abstract method.

20. (Original) The method of claim 1, wherein said generating step includes generating program code based on comments in a source file.

21. (Original) The method of claim 1, further comprising:

adding expected calls as instances of anonymous inner classes of the program; and

applying runtime introspection by a generated subclass to verify a sequence of expected calls.

22. (Original) The method of claim 1, wherein the component registers itself with a repository when the component is initially activated.

23. (Original) The method of claim 22, wherein the repository can be queried to determine components that are active.

24. (Currently amended) The method of claim 1, further comprising:

a~~A~~ computer-readable medium having processor-executable instructions for performing the method of claim 1.

25. (Currently amended) The method of claim 1, further comprising:

a~~A~~ downloadable set of processor-executable instructions for performing the method of claim 1.

26. (Currently amended) A system for dynamically generating program code based on declarative attributes, the system comprising:

a computer having a processor and memory;

a component module for creating a component from a program class based on adding a static field of type Component ~~component object~~ to the program class;

an attribute module for defining at least one declarative attribute specifying behavior to be added to the program class and associating said at least one attribute with the component, wherein said at least one declarative attribute comprises active metadata used to generate program code; and

a module for generating a subclass of the program class in response to instantiation of the component, the subclass including dynamically generated program code based on said at least one declarative attribute.

27. (Original) The system of claim 26, wherein the subclass adds tracing behavior to a program.

28. (Original) The system of claim 26, wherein the subclass is a subclass of an abstract class.

29. (Original) The system of claim 26, wherein said at least one declarative attribute includes active metadata, so as to provide a mechanism for generation of program code.

30. (Original) The system of claim 29, wherein said active metadata dynamically generates code for inclusion in the subclass of the program class.

31. (Original) The system of claim 26, wherein the attribute module provides for defining at least one attribute based on comments in source code of the program class.

32. (Original) The system of claim 31, further comprising:

a precomplier for precompiling a class containing static attributes from said comments.

33. (Original) The system of claim 32, wherein the module for generating loads the class containing static attributes before subclass generation.

34. (Original) The system of claim 31, further comprising:

an automated mapping between attribute syntax in comments and attribute syntax as expressed in generated program code.

35. (Original) The system of claim 26, wherein the attributed module provides for defining at least one attribute in a property file external to the program class.

36. (Original) The system of claim 35, further comprising:

a module for compiling a class containing dynamic attributes from said property file.

37. (Original) The system of claim 36, wherein the module for generating loads the class containing dynamic attributes before subclass generation when a component is instantiated.

38. (Original) The system of claim 35, further comprising:

an automated mapping between attribute syntax in a property file and attribute syntax as expressed in generated program code.

39. (Original) The system of claim 35, wherein attributes in a property file comprise property name and property value pairs.

40. (Original) The system of claim 26, wherein the attribute module provides for defining attributes for a superclass from which the program class inherits.

41. (Original) The system of claim 26, wherein the attribute module for provides for defining attributes for an interface from which the program class inherits.

42. (Original) The system of claim 41, wherein the module for generating generates an instance of a subclass to mock the behavior of the interface.

43. (Original) The system of claim 26, wherein the module for generating generates code for a non-abstract system body based on an attribute defined for an abstract method.

44. (Original) The system of claim 26, wherein the module for generating includes generating program code based on comments in a source file.

45. (Original) The system of claim 26, wherein the component registers itself with a repository when the component is initially activated.

46. (Original) The system of claim 45, wherein the repository can be queried to determine components that are active.

47. (Currently amended) A method for adding behavior to an application without access to application source code, the method comprising:

defining at least one attribute specifying declaratively behavior which is desired to be added to an application without access to the application source code, wherein said at least one attribute comprises active metadata used to generate code adding behavior to

the application;

storing said at least one attribute in a properties file external to the application;

creating a dynamic attributes class based on the properties file;

compiling the application and the dynamic attributes class; and

generating a subclass which includes dynamically generated code adding behavior to the application based on said at least one attribute.

48. (Original) The method of claim 47, wherein said defining step includes defining a particular attribute using active metadata, so as to provide a mechanism for generation of program code from said particular attribute.

49. (Original) The method of claim 48, wherein said active metadata dynamically generates code for inclusion in the subclass.

50. (Original) The method of claim 47, wherein said defining step includes defining an attribute for overriding a method of the application.

51. (Original) The method of claim 47, wherein said defining step includes defining an attribute for extending a method of the application.

52. (Original) The method of claim 47, further comprising:

loading the class containing dynamic attributes before generating the subclass.

53. (Original) The method of claim 47, further comprising:

defining an automated mapping between attribute syntax in the properties file and attribute syntax as expressed in generated program code.

54. (Original) The method of claim 47, wherein said at least one attribute in the properties file comprise property name and property value pairs.

55. (Original) The method of claim 47, wherein said creating step includes

creating a dynamic attributes class using a precompiler.

56. (Original) The method of claim 47, wherein said creating step includes creating a dynamic attributes class at runtime.

57. (Original) The method of claim 47, wherein said compiling step includes using a Java compiler (JAVAC).

58. (Original) The method of claim 47, wherein said generating step includes using a precompiler.

59. (Original) The method of claim 47, wherein said generating step includes using a runtime compiler.

60. (Currently amended) The method of claim 47, further comprising:

a~~A~~ computer-readable medium having processor-executable instructions for performing the method of claim 47.

61. (Currently amended) The method of claim 47, further comprising:

a~~A~~ downloadable set of processor-executable instructions for performing the method of claim 47.